Funzioni ricorsive

Funzioni che definiscono se stesse.

Abbiamo visto che una volta scritta una funzione, questa può essere chiamata da altre funzioni. Il linguaggio è costruito in modo tale che una funzione può chiamare anche sé stessa. Funzioni che che chiamano sé stesse si dicono ricorsive. Ad esempio la scalinata di Giacobbe può essere definita come uno scalino seguito da una scalinata di giacobbe. In Python:

```
def scalinata_di_Giacobbe():
    print(“Scalino”)
    scalinata_di_Giacobbe()
```

A prima vista si potrebbe pensare che questa procedura stampi una volta “Scalino” o al massimo 2 volte, invece continua a stampare “Scalino” finché c’è memoria disponibile o viene premuto il tasto `<Ctrl-c>`.
Come potremmo costruire una scala più umana? una scala di un certo numero di gradini? Prima di scrivere la procedura, scriviamo la definizione:

```
scala(n) = | se n=0 la scala è già finita
           | se n>0 è uno scalino seguito da una scala(n-1)
```

Utilizzando il comando `if` si può controllare la fine del lavoro della procedura. Scriviamo la procedura in un file:

```
def scala(n):
    if n == 0: return
    print(“scalino”)
    scala(n-1)
```

e proviamo il suo funzionamento: `<F5>`, poi nella shell di IDLE:

```
>>> scala(3)
scalino
scalino
scalino
```

Anche la procedura che stampa una parola triangolare può essere realizzata usando la ricorsione:

```
def triangolo(s):
    if s == "": return
    print(s)
    triangolo(s[1:])
```

Diciamo che una procedura ricorsiva è terminale se non vengono eseguite altre istruzioni dopo la chiamata ricorsiva. Le procedure viste sopra sono terminali.
Oltre alle procedure ricorsive, è anche possibile costruire funzioni ricorsive. Molte definizioni aritmetiche sono ricorsive e possono essere tradotte facilmente in funzioni ricorsive. Ad esempio una *potenza* che ha per esponente un numero naturale può essere definita in questo modo: un numero elevato all'esponente `enne` è uguale a `1` se `enne` è uguale a `0` altrimenti è uguale al numero per la potenza che ha esponente `enne - 1`.
Scritto in altro modo:

```
baseesp = | se esponente=0 allora la potenza è 1
          | se esponente>0 allora la potenza è base*baseesp-1
```

La definizione potrà sembrare piuttosto strana, ma tradotta in linguaggio di programmazione funziona:

```
def potenza(base, esponente):
    if esponente == 0:
        return 1
    else:
        return base * potenza(base, esponente - 1)
```

Altro esempio: il *fattoriale* del numero `enne` è `1` se il `enne` è `1`, altrimenti è `enne` per il `fattoriale` di `enne - 1`:

```
fatt. di n = | se n=0 allora il risultato è 1
             | se n>0 allora il risultato è n*fatt. di n-1
```

Un'altra definizione ricorsiva interessante è il metodo di Euclide per determinare il massimo comune divisore tra due numeri: se i due numeri sono uguali il massimo comune divisore è uno dei due, altrimenti è il massimo comune divisore tra il più piccolo e la differenza tra i due numeri:

```
                 | se n1=n2 il risultato è n1
macd(n1, n2) =   | se n1>n2 il risultato è macd(n1-n2, n2)
                 | se n2>n1 il risultato è macd(n2-n1, n1)
```

Data la definizione ricorsiva di una funzione, la sua traduzione in linguaggio Python risulta molto semplice.

Riassumendo

- Le procedure ricorsive con la condizione di terminazione permettono di ripetere dei blocchi di istruzioni.
- Una procedura ricorsiva si dice terminale se non vengono eseguite altre istruzioni dopo la chiamata ricorsiva.
- Molte definizioni matematiche possono essere espresse in forma ricorsiva e sono facilmente traducibili in Python.
- Le definizioni ricorsive sono interessanti perché risolvono un problema utilizzando i termini del problema stesso.

Prova tu

1. Cosa succede se nella procedura triangolo scambi tra di loro l'istruzione `print(s)` e la chiamata ricorsiva? Prima cerca di immaginarlo, poi controlla scrivendo quest'altra procedura e provandola.
2. Scrivi le funzioni che producono il fattoriale, e il massimo comun divisore.
3. Scrivi una procedura ricorsiva che stampi uno sotto l'altro gli elementi di una lista.
4. Scrivi una funzione ricorsiva che calcoli la somma dei numeri contenuti in una lista.

(da "Pylabmat" - https://pylabmat.readthedocs.io/it/latest/01_python/docs/source/117_ricorsione.html)¶

Aritmetica ricorsiva

Nel seguito è proposto un breve programma in cui sono utilizzate alcune funzioni ricorsive che definiscono operazioni aritmetiche su numeri interi positivi.

```
def add1(n,m):
    if m == 0:
        return n
    else:
        return add1(n+1, m-1)

def add2(n, m):
    if m == 0:
        return n
    else:
        return 1 + add2(n, m-1)

def molt(n, m):
    if n==0 or m == 0:
        return 0
    else:
        return n + molt(n, m-1)

def divInt(n, m):
    if n < m:
        return 0
    else:
        return 1 + divInt(n-m, m)

def pot(n, m):
    if m == 0:
        return 1
    else:
        return n * pot(n, m-1)

x = int(input("x = "))
y = int(input("y = "))
print(x, "+", y, "=", add1(x, y), "(addizione calcolata con add1)")
print(x, "+", y, "=", add2(x, y), "(addizione calcolata con add2)")
print(x, "x", y, "=", molt(x, y))
if y ==0:
    print("Impossibile dividere per 0")
else:
    print(x, ":", y, "=", divInt(x, y))
print(x, "**", y, "=", pot(x, y))
```